andrena OBJECTS

Experts in agile software engineering

Wenn der PO die Akzeptanztests schreibt – ATDD in der Praxis

Kathrin Ronellenfitsch und Dr. Rolf Schneeweiß

Agenda

1. Überblick und Setup aus der Praxis
2. Akzeptanztests automatisieren
3. Lebende Dokumentation erstellen
4. Den PO aktiv einbinden
5. Fazit

Agenda

1. **Überblick und Setup aus der Praxis**
2. Akzeptanztests automatisieren
3. Lebende Dokumentation erstellen
4. Den PO aktiv einbinden
5. Fazit

Praxisbeispiel

Ausgangslage in einem konkreten Projekt

- Cross-funktionales Feature-Team
- Kaum Unittests (um die 40%)
- Teilweise sehr alter Code (> 10 Jahre)
- Sehr viele Defects (> 100)
- Komplexe Businesslogik
- Hoher Druck neue Anforderungen umzusetzen und Defects abzuarbeiten

Einfach „weitermachen“ führt zwangsläufig zu neuen Fehlern

Die Testpyramide und die Praxis

- Wie es sein sollte:
  - Viele Unittests (> 80% Testabdeckung)
  - Automatisierte Integrationstests für alle Schnittstellen
  - Einige UI- und Akzeptanztests um das Zusammenspiel zu testen
- Die Praxis:
  - Kaum Unittests (< 50% Testabdeckung)
  - Keine automatisierten Integrationstests
  - Keinerlei Tests der kompletten Businesslogik

Praxisbeispiel

Legacy-Code stresst!

- Businesslogik komplex und teilweise sehr alt → Keiner versteht den Code
- Unittests quasi unmöglich nachzuziehen → Versuche enden immer wieder enttäuschend
- Hoher manueller Testaufwand → Trotzdem treten immer neue Defects auf
- Man traut sich nicht an alten Code → Es entstehen neue Welten (Code Duplications)

Features brauchen ewig, Stakeholder sind verärgert → Was nun?

Kurzer Einblick in cucumber

- Cucumber ist ein Framework um fachliche Akzeptanztests auszuführen
- Tests sind in lesbarer Sprache geschrieben:

```
1 Given today is Sunday
2 When I ask whether it's Friday yet
3 Then I should be told "Nope"
```

- Glue Code überführt die lesbaren Statements mit sogenannten step definitons in ausführbare Tests:

```
1 @When("I ask whether it's Friday yet")
2 public void i_ask_whether_it_s_Friday_yet() {
3 actualAnswer = IsItFriday.isItFriday(today);
4 }
```

Agenda

1. Überblick und Setup aus der Praxis
2. **Akzeptanztests automatisieren**
3. Lebende Dokumentation erstellen
4. Den PO aktiv einbinden
5. Fazit

cucumber im Build-Prozess

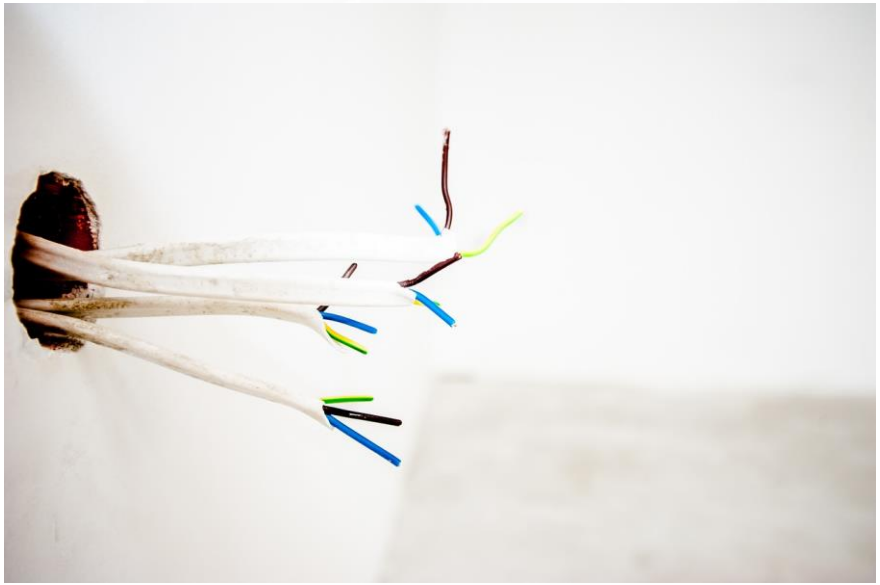
- Cucumber lässt sich leicht mit Gradle oder Maven benutzen
- Tests lassen sich wie Unittests ausführen

Cucumber lässt sich wie Unittests im Build-Prozess integrieren

File: gradle.build

```
1  task cucumber() {
2   dependsOn assemble, compileTestJava
3   doLast {
4    javaexec {
5     main = "cucumber.api.cli.Main"
6     classpath = configurations.cucumberRuntime +
                  sourceSets.main.output + sourceSets.test.output
8     args = ['--plugin', 'pretty', '--plugin', 'json:cucumber.json',
             '--glue', 'atdd', 'src/test/resources']
10   }
11  }
12 }
```

Mit cucumber einfach die API testen?



- Schnittstellen (APIs) lassen sich oft sehr einfach fachlich verstehen
- Tests mit Cucumber können helfen fachliche Fehler zu vermeiden
- Im Vergleich zu GUI-Tests sind Tests gegen eine API meist sehr schnell

Cucumber ist ideal um fachlich Schnittstellen zu testen

Praxisbeispiel

Legacy-Code mit Akzeptanztests testen

- Akzeptanztests sind bei Legacy-Code oft einfacher als Unittests nachzuziehen
- Höhere Testabdeckung → Refactorings werden möglich
- Das fachliche Verständnis des Codes steigt

Akzeptanztests reduzieren drastisch neue Fehler!

Ohne Automatisierung geht es nicht…

- Cucumber lässt sich einfach in Jenkins Build-Pipelines integrieren
- HTML-Reports lassen sich bequem per Plugin erstellen
- Akzeptanztests laufen bei jedem Build automatisch mit → Die Aufwände für manuelle Regressionstests lassen sich immens reduzieren

Cucumber und Jenkins sind ein perfektes Paar!

File: Jenkinsfile

```
1  pipeline {
2   agent any
3   stages {
4    stage('Acceptance tests') {
5     steps {
6      bat 'gradlew.bat cucumber'
7     }
8    }
9   }
10  post {
11    always {
12     cucumber buildStatus: 'UNSTABLE',
13     fileIncludePattern: '**/*.json',
14     trendsLimit: 10,
15     classifications: [['key': 'Version', 'value': 'Juni 2019']]
16    }
17  }
18 }
```


Praxisbeispiel

Defects mit Akzeptanztests bekämpfen

- Zu jedem fachlichen Defect wird wenigstens ein Akzeptanztest angelegt
- Alles erfolgt in enger Abstimmung mit dem PO → Das Verständnis für die Fachlichkeit steigt
- Der Defect wird gefixt
- Der Akzeptanztest läuft zukünftig in der Regression immer mit

Automatische Akzeptanztests sind nachhaltig!

Agenda

1. Überblick und Setup aus der Praxis
2. Akzeptanztests automatisieren
3. **Lebende Dokumentation erstellen**
4. Den PO aktiv einbinden
5. Fazit

Oje, die Testdokumentation…

- Stakeholder verlangen eine ausführliche Dokumentation → Meist sehr hoher Aufwand
- Dokumentationen sind oft nicht auf dem aktuellen Stand
- Cucumber ermöglicht dagegen sehr einfach eine „lebende“ Dokumentation

Mit Cucumber macht sich die Testdokumentation quasi von alleine!

File: Stepdefs.java

```
1  public class Stepdefs {
2   private LocalDate date;
3   private Products product;
4   private Amount actualAmount;
5
6   @After
7   public void attachHtml(Scenario scenario) {
8        String htmlAttachment =
              HTMLGenerator.generateHTMLFileAsString(product, date, actualAmount);
10       byte[] htmlByteArray = htmlAttachment.getBytes(StandardCharsets.UTF_8);
11       scenario.embed(htmlByteArray, "text/html");
12  }
13 }
```


Praxisbeispiel

Dokumentation nervt, allerdings…

- Bringt Credibility
- Sorgt für zufriedene Stakeholder
- Schafft Transparenz
- Ermöglicht häufigere Releases
- Tests müssen zum Alltag gehören → Nur wenn das Mindset stimmt, erhöht sich die Qualität

Transparenz schafft Vertrauen!

Agenda

1. Überblick und Setup aus der Praxis
2. Akzeptanztests automatisieren
3. Lebende Dokumentation erstellen
4. **Den PO aktiv einbinden**
5. Fazit

Akzeptanztests von den Entwicklern…

- Den Entwicklern fehlt oft das fachliche Know-How
- Viele Rückfragen beim PO nötig → Zeitverlust und Defokussierung
- Zusammenarbeit stärken → Vielleicht müssen die Entwickler nicht alles alleine machen

…vielleicht sollte jemand anderes die Akzeptanztests schreiben?

Praxisbeispiel

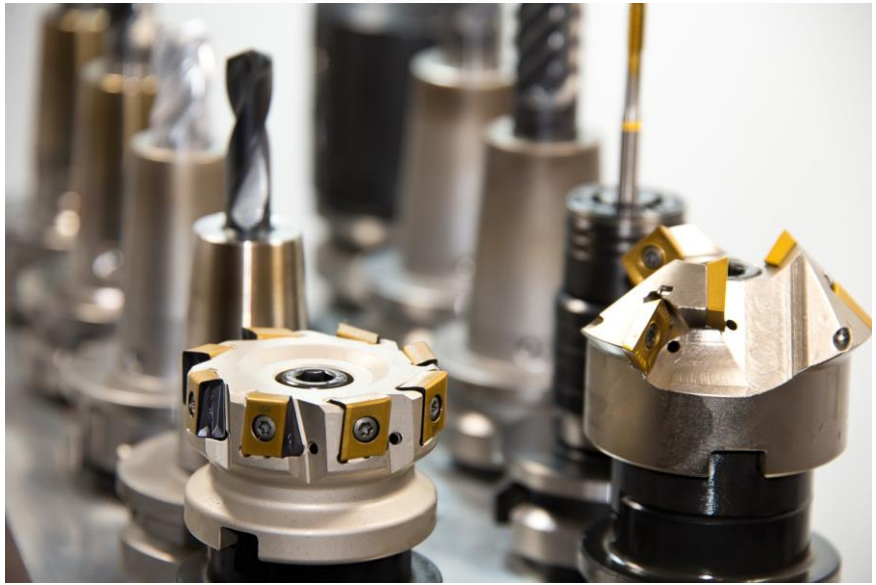
Der PO als Tester

- Der PO oder Business Analyst (BA) kennt die Fachlichkeit → Perfekt um Akzeptanztests zu schreiben
- Test first ist einfach möglich, da die Fachlichkeit vor der Umsetzung da ist
- Dem PO, bzw. BA müssen die passenden Tools gegeben werden

PO und Entwickler arbeiten zusammen an einer Sache!

Praxisbeispiel

Tools für den PO

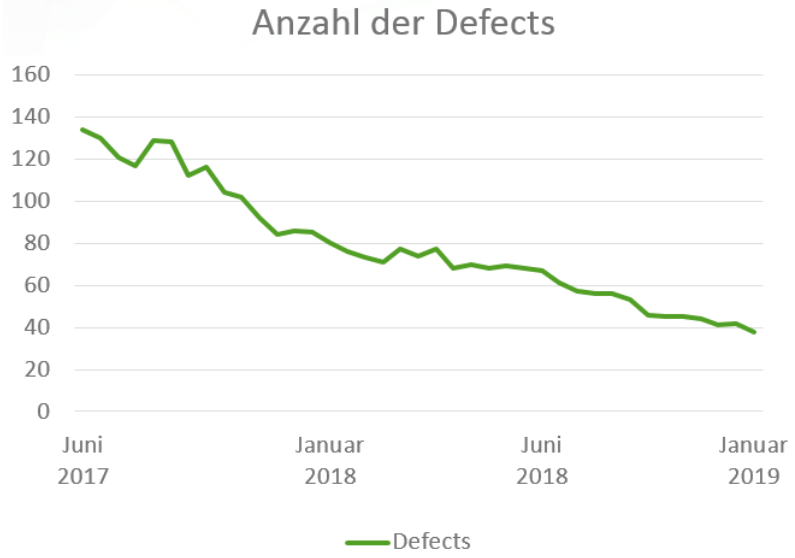


- Specification by example als Grundwerkzeug
- Fehlendes fachliches Verständnis des Teams wird transparent
- PO tat sich mit Gherkin schwer → Excel als Tool hat gut funktioniert
- Zugang zu Git und Jenkins für den PO waren super

Mit den richtigen Tools klappt alles besser

Praxisbeispiel

Hat das denn was gebracht?



- Die Anzahl an Fehlern im konkreten Projekt konnte deutlich reduziert werden
- Viele Faktoren sind wichtig, Akzeptanztests sind einer
- Wenn die Richtung und der Spirit stimmen, geht alles einfacher

Akzeptanztests helfen die Qualität zu steigern!

Agenda

1. Überblick und Setup aus der Praxis
2. Akzeptanztests automatisieren
3. Lebende Dokumentation erstellen
4. Den PO aktiv einbinden
5. **Fazit**

Fazit: Legacy, Feature-Druck, Stress – was jetzt?

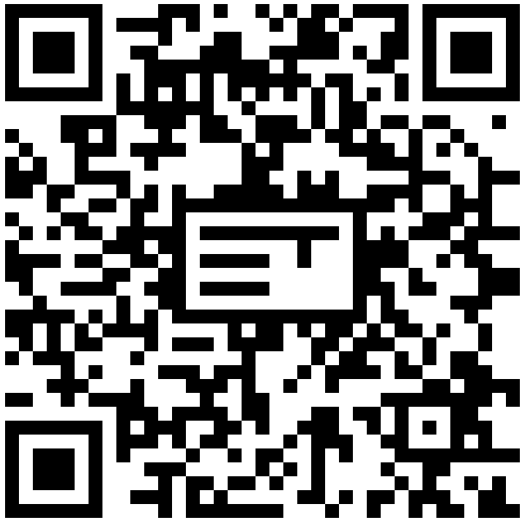
Akzeptanztests helfen und sind einfacher als man denkt!

Bilder von Pixabay

Bitte geben Sie uns jetzt Ihr Feedback!

Wenn der PO die Akzeptanztests schreibt
– ATDD in der Praxis

Dr. Rolf Schneeweiß, Kathrin Ronellenfitsch



Nächste Vorträge in diesem Raum

14:30 Webservices testen als Mob – Das Warum, das Was und das Wie, *Mario Kühne*

15:45 Property-based Testing in Java, *Johannes Link*